



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/687,233	10/15/2003	Martin Runte	6570P015	8107

8791 7590 11/03/2006

BLAKELY SOKOLOFF TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1030

EXAMINER

CHEN, QING

ART UNIT	PAPER NUMBER
----------	--------------

2191

DATE MAILED: 11/03/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary

Application No.

10/687,233

Applicant(s)

RUNTE ET AL.

Examiner

Qing Chen

Art Unit

2191

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 15 October 2003.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-38 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-38 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☒ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 15 October 2003 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.
- Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
- Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
- ☐ Certified copies of the priority documents have been received.
 - ☐ Certified copies of the priority documents have been received in Application No. _____.
 - ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) ☐ Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date _____
- 4) ☐ Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____
- 5) ☐ Notice of Informal Patent Application
- 6) ☐ Other: _____

DETAILED ACTION

1. This is the initial Office action based on the application filed on October 15, 2003.

Claims 1-38 are currently pending and have been considered below.

Specification

2. The abstract of the disclosure is objected to because it contains a paragraph number.

Correction is required. See MPEP § 608.01(b) and 37 CFR § 1.72(b).

3. The disclosure is objected to because of the following informalities:

- The specification contains the following typographical errors:
 - The terminology "Frozen objects" should presumably read -- Frozen objects table -- in page 4, paragraph [0018]. Applicant is advised to make the correction in order to keep the terminology consistent throughout the specification.
 - Reference number "210" should be changed to "215" in page 8, paragraph [0025].

Appropriate correction is required.

Claim Objections

4. **Claims 1-6, 23, and 32-38** are objected to because of the following informalities:

- **Claims 1, 6, and 32** contain a typographical error: the word "and" should be added after the second-to-last limitation.

- **Claims 1, 32, and 36** recite the limitation “the objects.” The Examiner subsequently interprets this limitation as reading “the software objects” for the purpose of providing it with proper explicit antecedent basis.

- **Claims 2-5, 33-35, 37, and 38** depend on Claims 1, 32, and 36, respectively.

Therefore, these claims suffer the same deficiency as their respective parent claims.

- **Claim 23** contains the following typographical errors:

- The term “computer-implemented” in “the computer-implemented method” should be deleted, since all other dependent claims of the group refer to the parent claim as “the method.”

- The phrase “... at a time of a last global compatibility check” should presumably read “... at the time of a last global compatibility check.”

- **Claims 34 and 35** contain a typographical error: “wherein the method” should be deleted. Applicant is advised to make the correction in order to keep the claim language consistent with Claim 33.

- **Claim 37** contains a typographical error: “further comprising” should presumably read -- wherein --.

- **Claims 37 and 38** contain a typographical error: “further comprise” should presumably read -- further comprising --.

Appropriate correction is required.

Claim Rejections - 35 USC § 112

5. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

6. **Claim 10** is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 10 recites the limitation “a small number.” The term “a small number” is a relative term, which renders the claim indefinite. The term “a small number” is not defined by the claim nor does the specification provide a standard for ascertaining the requisite degree and one of ordinary skill in the art would not be able to reasonably determine the scope of the invention. In the interest of compact prosecution, the Examiner subsequently does not give any patentable weight to this limitation for the purpose of further examination.

Claim Rejections - 35 USC § 101

7. 35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

8. **Claims 1-21 and 26-38** are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

Claims 1, 3-5, 26, 32, 34-36, and 38 recite the limitation of allowing the change if the change is considered to be compatible. However, in the case of where the “if” condition does not occur, then the claims do not produce a useful, concrete, and tangible result because the claims do not recite any step or means for an “else” condition. Therefore, the claims do not produce a practical application of the § 101 judicial exception.

The result of **Claims 1, 3, 32, and 35** is directed to the act of “allowing,” which does not appear to be a tangible result so as to constitute a practical application of the idea. The act of “allowing” is merely a thought or an abstract idea and does not appear to produce a tangible result even if the step of allowance does occur, since the result of that allowance is not conveyed in the real world. The result is an allowance, which is neither used in a disclosed practical application nor made available for use in a disclosed practical application. It also does not appear that the usefulness of the allowance can be realized from the claimed steps to support a disclosed specific, substantial, and credible utility so as to produce a useful result.

Therefore, the claims do not meet the statutory requirement of 35 U.S.C. § 101, since the claims are not directed to a practical application of the § 101 judicial exception producing a result tied to the physical world.

The results of **Claims 2, 6-21, 33, and 37** are directed to the act of “determining,” which do not appear to be a tangible result so as to constitute a practical application of the idea.

Art Unit: 2191

Therefore, these claims are rejected for the same reason set forth in the rejections of Claims 1, 3, 32, and 35 above.

Claims 26-31 are directed to apparatus of functional descriptive material *per se*, and hence non-statutory. The recited components of the claims can reasonably be interpreted as computer program modules—software *per se*. Also, the specification discloses that many of the features and techniques may be implemented in software (*see Specification – Paragraph [0040]*). Therefore, the claims constitute computer programs representing computer listings *per se*. Such descriptions or expressions of the programs are not physical “things.” They are neither computer components nor statutory processes, as they are not “acts” being performed. Such claimed computer programs do not define any structural and functional interrelationships between the computer program and other claimed elements of a computer, which permit the computer program’s functionality to be realized. In contrast, a claimed computer-readable medium encoded with a computer program is a computer element, which defines structural and functional interrelationships between the computer program and the rest of the computer, that permits the computer program’s functionality to be realized, and is thus statutory. See *Lowry*, 32 F.3d at 1583-84, 32 USPQ2d at 1035.

Claims 36-38 contain “means for” limitations. However, it is noted that the specification does not disclose any specific corresponding structure or equivalents thereof. Therefore, these claim limitations can reasonably be interpreted as computer program modules—software *per se*. The claims are directed to apparatus of functional descriptive material *per se*, and hence non-

statutory. Therefore, these claims are rejected for the same reason set forth in the rejections of Claims 26-31 above.

Claim Rejections - 35 USC § 102

9. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

10. **Claims 1-4, 6-20, 22-27, 29-34, 36, and 37** are rejected under 35 U.S.C. 102(e) as being anticipated by **Carter et al.** (US 6,519,767).

As per **Claim 1**, **Carter et al.** disclose:

- detecting a change introduced into a software object of a first software subsystem, wherein the software object is used by software objects of a second software subsystem (*see Column 8: 12-17, "Version compatibility analyzer 70 utilizes a process 76 illustrated in FIG. 6 and described below to detect any modifications ("version incompatible differences") from existing object server 66 that prevent compiler 52 from building new object server 64 so as to be version compatible with existing object server 66."*); and
- allowing the change if the change is compatible with the software objects of the second software subsystem without introducing any changes into the software objects of the

Art Unit: 2191

second software subsystem (*see Column 8: 21-25, "When no version incompatible differences are detected by version compatibility analyzer 70, compatible object server generator 72 generates the interface related information in process 78 so as to be version compatible with existing object server 66."*).

As per **Claim 2**, the rejection of **Claim 1** is incorporated; and Carter et al. further disclose:

- determining whether the change is predefined as compatible (*see Column 13: 10-18, "For example, the following edits do not prevent new object server 64 from being version compatible with existing object server 66: changing code within a member, changing the position of a member in a class, and adding a new member to a class."*).

As per **Claim 3**, the rejection of **Claim 2** is incorporated; and Carter et al. further disclose:

- allowing the change if the change is predefined as compatible (*see Column 13: 10-13, "... edits which do not change any interface supported by existing object server 66 can be made to source data 60 without making new object server 64 incompatible."*).

As per **Claim 4**, the rejection of **Claim 3** is incorporated; and Carter et al. further disclose:

Art Unit: 2191

- issuing a message that the change is not allowed if the change is not predefined as compatible (*see Column 14: 25-28, "... user interface 59 notifies the user that new object server 64 cannot be compatible, such as by displaying a dialog box with such a message."*).

As per **Claim 6**, Carter et al. disclose:

- identifying a subset of software objects of a first software subsystem and declaring the subset of software objects frozen (*see Column 8: 30-37, "In the existing version object server 92, clock object 94 has three members 96-98 (FIG. 3A), respectively named "SetTime," "GetHour," and "GetMinute," as well as properties 100 which include integer values named "hour," and "minute." In the revised version object server 92', clock object 94' is changed by adding a new member 99, named "SetAlarm," along with new integer values, "alarmHour," and "alarmMinute" to properties 100'. A client application 101 (FIG. 3) manipulates or controls clock object 94 or 94' by calling members 96-99."*); and

- detecting a change introduced into a frozen software object from the subset of software objects, and prior to allowing the change determining whether the change is compatible with a second software subsystem (*see Column 8: 12-17, "Version compatibility analyzer 70 utilizes a process 76 illustrated in FIG. 6 and described below to detect any modifications ("version incompatible differences") from existing object server 66 that prevent compiler 52 from building new object server 64 so as to be version compatible with existing object server 66."*).

As per **Claim 7**, the rejection of **Claim 6** is incorporated; and Carter et al. further disclose:

Art Unit: 2191

- wherein the subset of software objects declared frozen includes software objects of the first software subsystem that are used by the second software subsystem (*see Column 8: 30-37, "In the revised version object server 92', clock object 94' is changed by adding a new member 99, named "SetAlarm," along with new integer values, "alarmHour," and "alarmMinute" to properties 100'. A client application 101 (FIG. 3) manipulates or controls clock object 94 or 94' by calling members 96-99."*).

As per **Claim 8**, the rejection of **Claim 7** is incorporated; and Carter et al. further disclose:

- wherein frozen objects are classified to include released objects and restricted objects (*see Column 8: 30-37, "In the revised version object server 92', clock object 94' is changed by adding a new member 99, named "SetAlarm," along with new integer values, "alarmHour," and "alarmMinute" to properties 100'. A client application 101 (FIG. 3) manipulates or controls clock object 94 or 94' by calling members 96-99."*).

As per **Claim 9**, the rejection of **Claim 8** is incorporated; and Carter et al. further disclose:

- wherein the released objects include objects that are used by the second software subsystem without restrictions (*see Column 8: 30-37, "In the revised version object server 92', clock object 94' is changed by adding a new member 99, named "SetAlarm," along with new integer values, "alarmHour," and "alarmMinute" to properties 100'. A client application 101 (FIG. 3) manipulates or controls clock object 94 or 94' by calling members 96-99."*).

As per **Claim 10**, the rejection of **Claim 8** is incorporated; and Carter et al. further disclose:

- wherein the restricted objects include objects that are used by a small number of objects of the second software subsystem (*see Column 8: 30-37, "In the revised version object server 92', clock object 94' is changed by adding a new member 99, named "SetAlarm," along with new integer values, "alarmHour," and "alarmMinute" to properties 100'. A client application 101 (FIG. 3) manipulates or controls clock object 94 or 94' by calling members 96-99. "*).

As per **Claim 11**, the rejection of **Claim 8** is incorporated; and Carter et al. further disclose:

- wherein an identification of recent changes introduced into a restricted object is provided when objects of the second software subsystem request new usage of the restricted object (*see Column 8: 55-59, "In this vtable, ITime₁ interface 110 includes pointers to members 96-98 of clock object 94. The members 96-98 can then be accessed by code in client application 101 based on their particular offset within the vtable. "*).

As per **Claim 12**, the rejection of **Claim 8** is incorporated; and Carter et al. further disclose:

- wherein classification of the frozen objects is based on a number of times a frozen object is used by the second software subsystem (*see Column 9: 1-6, "AddRef and Release*

Art Unit: 2191

functions 125-126 include code for OLE 2 to track the number of references from client applications currently using an instance of clock object so that memory allocated to such instance can be released when the instance is no longer in use.”).

As per **Claim 13**, the rejection of **Claim 6** is incorporated; and Carter et al. further disclose:

- wherein a software object is a function module (*see Column 7: 13-16, “Integrated development environment 50 further includes a source editor 56 which provides conventional editing tools for a user to enter and modify programs, including programs which are to be compiled into object servers.”).*

As per **Claim 14**, the rejection of **Claim 6** is incorporated; and Carter et al. further disclose:

- wherein a software object is a data structure (*see Column 7: 27-31, “Source data 60 of the program can be stored by source editor 56 in the form of a data stream representation of the programming language statements, or more preferably a data stream consisting of a tokenized representation of the programming language statements.”).*

As per **Claim 15**, the rejection of **Claim 13** is incorporated; and Carter et al. further disclose:

- wherein the software object includes an environment of the function module (*see Column 7: 13-16, “Integrated development environment 50 further includes a source editor 56*

Art Unit: 2191

which provides conventional editing tools for a user to enter and modify programs, including programs which are to be compiled into object servers."

As per **Claim 16**, the rejection of **Claim 6** is incorporated; and Carter et al. further disclose:

- wherein a software object includes a class and an environment of the class (see Column 7: 25-27, "In the Visual Basic language system, each project can include one or more class modules which each define an object.").

As per **Claim 17**, the rejection of **Claim 6** is incorporated; and Carter et al. further disclose:

- wherein a software object includes an interface and an environment of the interface (see Column 7: 25-27, "In the Visual Basic language system, each project can include one or more class modules which each define an object.").

As per **Claim 18**, the rejection of **Claim 6** is incorporated; and Carter et al. further disclose:

- wherein a software object includes a program and an environment of the program (see Column 7: 23-25, "... each program is entered by the user in the form of a "project" in the Visual Basic language system.").

As per **Claim 19**, the rejection of **Claim 6** is incorporated; and Carter et al. further disclose:

- wherein the detecting the change comprises automatically monitoring development of software code (*see Figure 6; Column 13: 36-46, "In process 76, version compatibility analyzer 70 determines version compatibility with existing object server 66 by comparing source data 60 to type information stored in existing object server's type library 150."*).

As per **Claim 20**, the rejection of **Claim 6** is incorporated; and Carter et al. further disclose:

- wherein the determining whether the change is compatible comprises determining whether there is a predefined declaration of compatibility of the change (*see Column 13: 10-18, "For example, the following edits do not prevent new object server 64 from being version compatible with existing object server 66: changing code within a member, changing the position of a member in a class, and adding a new member to a class."*).

As per **Claim 22**, Carter et al. disclose:

- performing a global compatibility check of software objects of a first software subsystem by determining whether any changes were introduced into a subset of the software objects of the first software subsystem since the time of a last compatibility check, wherein the introduced changes were introduced without obtaining prior approval (*see Figure 6; Column 12: 62-67, "FIG. 6 shows version compatibility analysis process 76 (FIG. 6) comprising steps 160-167 which is utilized by compiler 52 (FIG. 2) in version compatibility analyzer 70 (FIG. 2) to*

Art Unit: 2191

determine whether new object server 64 (FIG. 2) can be version compatible with existing object server 66 (FIG. 2).");

- identifying software objects of a second software subsystem affected by an unapproved change, wherein the affected software objects of the second software system are software objects using at least one software object of the subset of the software objects of the first software system (*see Figure 6: 164 and 165; Column 7: 60-65, "The user also can select with user input 58 to have compiler 52 compile source data 60 utilizing the process according to the present invention for automatically building a version compatible object server, or in the conventional manner (without the version compatible object server building process)."; Column 8: 3-8, "If no file name is specified in the Compatible Object Application field (such as when compiling the first version of existing object server 66), compiler 52 compiles source data 60 conventionally into an object server without the automatic version compatible object server building process."; Column 14: 33-38, "As indicated at steps 164 and 165, when new object server 64 is determined from the step 161 comparison to have added any class or any public member to a class over existing object server 66, new object server 64 is determined by version compatibility analyzer 70 to be version compatible with existing object server 66."); and*

- issuing a notice of possible incompatibility between affected software objects and software objects including the unapproved change (*see Column 14: 25-28, "... user interface 59 notifies the user that new object server 64 cannot be compatible, such as by displaying a dialog box with such a message.").*

As per **Claim 23**, the rejection of **Claim 22** is incorporated; and Carter et al. further disclose:

- wherein the performing a global compatibility check comprises comparing a current version of software code with a version of the software code at the time of a last global compatibility check (*see Column 12: 62-67, "FIG. 6 shows version compatibility analysis process 76 (FIG. 6) comprising steps 160-167 which is utilized by compiler 52 (FIG. 2) in version compatibility analyzer 70 (FIG. 2) to determine whether new object server 64 (FIG. 2) can be version compatible with existing object server 66 (FIG. 2)."*).

As per **Claim 24**, the rejection of **Claim 22** is incorporated; and Carter et al. further disclose:

- wherein the subset of the software objects includes frozen software objects (*see Column 8: 30-37, "In the revised version object server 92', clock object 94' is changed by adding a new member 99, named "SetAlarm," along with new integer values, "alarmHour," and "alarmMinute" to properties 100'. A client application 101 (FIG. 3) manipulates or controls clock object 94 or 94' by calling members 96-99."*).

As per **Claim 25**, the rejection of **Claim 24** is incorporated; and Carter et al. further disclose:

- wherein the frozen software objects include objects of the first software subsystem used by objects of the second software subsystem (*see Column 8: 30-37, "In the revised version object server 92', clock object 94' is changed by adding a new member 99, named "SetAlarm,"*

Art Unit: 2191

along with new integer values, "alarmHour," and "alarmMinute" to properties 100'. A client application 101 (FIG. 3) manipulates or controls clock object 94 or 94' by calling members 96-99. ").

As per **Claim 26**, Carter et al. disclose:

- a changes monitor to automatically detect a change introduced into a software object of a first software subsystem, wherein the software object is used by objects of a second software subsystem, and the changes monitor to allow the change if the change is compatible with the objects of the second software subsystem without introducing any changes into the objects of the second software subsystem (*see Figure 2: 52; Column 8: 12-17, "Version compatibility analyzer 70 utilizes a process 76 illustrated in FIG. 6 and described below to detect any modifications ("version incompatible differences") from existing object server 66 that prevent compiler 52 from building new object server 64 so as to be version compatible with existing object server 66."*; *Column 8: 21-25, "When no version incompatible differences are detected by version compatibility analyzer 70, compatible object server generator 72 generates the interface related information in process 78 so as to be version compatible with existing object server 66."*); and
- an error notification module to notify a software developer introducing the change into the object of the first software subsystem of a not allowed change if the change is incompatible (*see Column 14: 25-28, "... user interface 59 notifies the user that new object server 64 cannot be compatible, such as by displaying a dialog box with such a message."*).

As per **Claim 27**, the rejection of **Claim 26** is incorporated; and Carter et al. further disclose:

- wherein the changes monitor to allow the change if the change is compatible comprises the changes monitor to determine whether there is a predefined declaration of compatibility of the change (*see Column 13: 10-18, "For example, the following edits do not prevent new object server 64 from being version compatible with existing object server 66: changing code within a member, changing the position of a member in a class, and adding a new member to a class."*).

As per **Claim 29**, the rejection of **Claim 26** is incorporated; and Carter et al. further disclose:

- a master system including the changes monitor to detect a change introduced into a software object of a first software subsystem from a plurality of software subsystems (*see Figure 2: 50; Column 7: 1-5, "FIG. 2 is a block diagram of a programming language system or integrated development environment 50 which provides a compiler 52 for building version compatible object servers according to a preferred embodiment of the invention."*).

As per **Claim 30**, the rejection of **Claim 26** is incorporated; and Carter et al. further disclose:

- wherein the first software subsystem is located at a server (*see Column 6: 16-18, "Referring to FIG. 1, an operating environment for the preferred embodiment of the present invention is a computer system 20 ..."*).

As per **Claim 31**, the rejection of **Claim 26** is incorporated; and Carter et al. further disclose:

- wherein the second software subsystem is located at a client (*see Column 6: 16-18, "Referring to FIG. 1, an operating environment for the preferred embodiment of the present invention is a computer system 20 ... "*).

Claims 32-34 are article of manufacture claims corresponding to the method claims above (Claims 1, 2, and 4) and, therefore, are rejected for the same reason set forth in the rejections of Claims 1, 2, and 4.

As per **Claim 36**, Carter et al. disclose:

- means for detecting a change introduced into a software object of a first software subsystem, wherein the software object is used by software objects of a second software subsystem (*see Column 8: 12-17, "Version compatibility analyzer 70 utilizes a process 76 illustrated in FIG. 6 and described below to detect any modifications ("version incompatible differences") from existing object server 66 that prevent compiler 52 from building new object server 64 so as to be version compatible with existing object server 66. "*);
- means for allowing the change if the change is compatible with the software objects of the second software subsystem without introducing any changes into the software objects of the second software subsystem (*see Column 8: 21-25, "When no version incompatible differences are detected by version compatibility analyzer 70, compatible object server generator*

Art Unit: 2191

72 generates the interface related information in process 78 so as to be version compatible with existing object server 66. "); and

- means for issuing a notice of a not allowed change if the change is not compatible (see Column 14: 25-28, "... user interface 59 notifies the user that new object server 64 cannot be compatible, such as by displaying a dialog box with such a message. ").

As per **Claim 37**, the rejection of **Claim 36** is incorporated; and Carter et al. further disclose:

- wherein means for allowing the change further comprise means for determining whether the change is predefined as compatible (see Column 13: 10-18, "For example, the following edits do not prevent new object server 64 from being version compatible with existing object server 66: changing code within a member, changing the position of a member in a class, and adding a new member to a class. ").

Claim Rejections - 35 USC § 103

11. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

12. **Claims 5, 21, 28, 35, and 38** are rejected under 35 U.S.C. 103(a) as being unpatentable over Carter et al. (US 6,519,767) in view of Hiller et al. (US 6,658,659).

As per **Claim 5**, the rejection of **Claim 4** is incorporated; however, Carter et al. do not disclose:

- allowing the change if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible.

Hiller et al. disclose allowing the change if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible (*see Column 11: 21-25, "... allow a programmer to designate acceptable compatibility according to a minimum version number or a version number corresponding to a minimum date of release."*).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hiller et al. into the teaching of Carter et al. to include allowing the change if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible. The modification would be obvious because one of ordinary skill in the art would be motivated to select a suitable version of a program to avoid version incompatibility problems as new versions are released (*see Hiller et al. – Column 13: 6-12*).

As per **Claim 21**, the rejection of **Claim 7** is incorporated; however, Carter et al. do not disclose:

- wherein the determining whether the change is compatible comprises determining whether an expert declared the change compatible.

Hiller et al. disclose wherein the determining whether the change is compatible comprises determining whether an expert declared the change compatible (*see Column 11: 21-25, "... allow a programmer to designate acceptable compatibility according to a minimum version number or a version number corresponding to a minimum date of release."*).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hiller et al. into the teaching of Carter et al. to include wherein the determining whether the change is compatible comprises determining whether an expert declared the change compatible. The modification would be obvious because one of ordinary skill in the art would be motivated to select a suitable version of a program to avoid version incompatibility problems as new versions are released (*see Hiller et al. – Column 13: 6-12*).

As per **Claim 28**, the rejection of **Claim 26** is incorporated; however, Carter et al. do not disclose:

- wherein the change monitor to allow the change if the change is compatible comprises the changes monitor to determine if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible.

Hiller et al. disclose wherein the change monitor to allow the change if the change is compatible comprises the changes monitor to determine if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible (*see Column 11: 21-25, "... allow a programmer to designate*

Art Unit: 2191

acceptable compatibility according to a minimum version number or a version number corresponding to a minimum date of release.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hiller et al. into the teaching of Carter et al. to include wherein the change monitor to allow the change if the change is compatible comprises the changes monitor to determine if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible. The modification would be obvious because one of ordinary skill in the art would be motivated to select a suitable version of a program to avoid version incompatibility problems as new versions are released (*see Hiller et al. – Column 13: 6-12*).

Claim 35 is rejected for the same reason set forth in the rejection of Claim 5.

As per **Claim 38**, the rejection of **Claim 36** is incorporated; however, Carter et al. do not disclose:

- wherein means for allowing the change further comprising means for allowing the change if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible.

Hiller et al. disclose wherein means for allowing the change further comprising means for allowing the change if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible (*see Column 11:*

Art Unit: 2191

21-25, "... allow a programmer to designate acceptable compatibility according to a minimum version number or a version number corresponding to a minimum date of release.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hiller et al. into the teaching of Carter et al. to include wherein means for allowing the change further comprising means for allowing the change if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible. The modification would be obvious because one of ordinary skill in the art would be motivated to select a suitable version of a program to avoid version incompatibility problems as new versions are released (see Hiller et al. – Column 13: 6-12).

Conclusion

13. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

A. Gerken et al. (US 5,729,744) disclose an improved method and system for controlling versions of objects that makes use of the change control history of the objects.

B. Soni et al. (US 5,860,007) disclose a method for automating the process of making modular architectural and source code changes to system software, thereby minimizing the time and the amount of human intervention necessary to complete such changes, as well as the number of errors usually resulting from implementing such changes.

C. McIntyre (US 6,415,435) discloses a method and apparatus for versioning and identifying compatibility between classes in an object-oriented hierarchy.

D. **Hurley et al.** (US 6,678,882) disclose a collaboration model, which can support the introduction of new object types into deployed network environments.

E. **Te'eni et al.** (US 6,725,452) disclose a method for resolving dependency conflicts across diverse sets of functional entities while installing or removing specific operative elements in a computing environment.

F. **Spring** (US 6,971,093) discloses maintaining compatibility between a software core module that is developed by one group and one or more separately developed domain specific interacting modules that interface with the core module.

G. **Kalthoff et al.** (US 7,031,787) disclose methods and apparatus, including computer program products, for managing changes in a product creation process.

H. **Kramer** (US 7,076,764) discloses a system and method for determining and supporting dependencies between objects or modules in a software development project.

I. **Sundararajan** (US 2002/0072928) discloses a method and system for ensuring compatibility between a plurality of components related to an organization's production environment.

J. **Harrison et al.** (US 2002/0078262) disclose a system and methods that provide compatibility across multiple versions of a software system, such as an execution engine or run-time system.

K. **Massaro et al.** (US 2004/0230952) disclose marking changes in versions of files based on a region and a threshold.


Art Unit: 2191

Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Qing Chen whose telephone number is 571-270-1071. The Examiner can normally be reached on Monday through Thursday from 7:30 AM to 4:00 PM. The Examiner can also be reached on alternate Fridays.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Wei Zhen, can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).


Supervising Patent Examiner

QC / QC
October 26, 2006